DOCUMENT RESUME

ED 267 810 IR 051 471

AUTHOR Duffy, Thomas M.; Langston, M. D.
TITLE On-Line Help: Design Issues for Authoring Systems. CDC Technical Report No. 18.
INSTITUTION Carnegie-Mellon Univ., Pittsburgh, PA. Communications Design Center.
PUB DATE Nov 85
NOTE 29p.
PUB TYPE Information Analyses (070) -- Reports - Descriptive (141)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Authoring Aids (Programing); Communication Research; *Computer Assisted Instruction; *Computer Software; Guidelines; Information Needs; *Online Systems; Position Papers; Systems Development
IDENTIFIERS *Help Systems

ABSTRACT

Part of the Communications Design Center's (CDC) on-going research on communication and document design, this report emphasizes the importance of a multi-level computer help system that goes beyond simple command descriptions to include assistance with goals and strategies within the application domain--in this case, automated authoring. The help system should be dynamic to at least some degree, basing its presentation of help options on the user's current activity. It is also suggested that an adequate help system would include both a mechanism for controlling the amount of information presented on a help request and an interactive facility to allow users to try out different features in a procedural fashion. Finally, the textual information contained in the help database must be carefully designed and structured to make the necessary assistance both accessible and understandable for maximum user efficiency. An introductory discussion of computer-based instruction (CBI) is followed by five sections: Defining the Audience; Prototype Help Systems; Factors in the Design of Help (Dynamism and Access); Information Requirements (Level of Explanation, Amount of Information Presented, Level of Detail, Categories of Information, and Tutorials); and An Evaluation of a Help System. A summary and a 42-item reference list complete the report. (THC)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
OERI
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)
X This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official position or policy.

ED267810

CDC Technical Report No. 18

ON-LINE HELP: Design Issues for Authoring Systems

Thomas M. Duffy and M.D. Langston

November 1985

IR051471

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Nicolette Canterna

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

The Communications Design Center was established in 1979 to solve communication problems in business, industry, government, and the professions.

The goals of the Center are:

- to encourage interdisciplinary approaches to the solution of visual and verbal communication problems;
- to promote cooperation between communication researchers at the University and those outside the University who have practical interests in communication problems;
- to provide resources and an organizational setting for research in visual and verbal communication;
- to train professional writers; researchers, and teachers;
- to promote the development of educational materials for communication programs on and off the CMU campus.

This report is one of a series from our on-going research on communication and document design. If you have questions or comments on this report, or if you would like to find out more about the activities of the CDC, contact:

Director
Communications Design Center
Baker Hall 160
Carnegie-Mellon University
Pittsburgh, PA 15213
(412) 578-2906

Online Help:

Design Issues for Authoring Systems[1]

Thomas M. Duffy and M. Diane Langston
Communications Design Center, Carnegie-Mellon University

Introduction

Computer-based instruction (CBI) has been heralded as an educational panacea since the early applications of the 1950's. Many believed that computers would see widespread instructional use, presenting high quality educational materials with pacing and feedback geared to the individual student. In fact CBI has not been so widely used. At first it seemed that the high cost of computer systems prevented users from investing in CBI. In recent years, however, hardware costs have dropped dramatically and microcomputers have invaded homes, schools, and businesses. Users are exploiting computer resources for database management, accounting, and word processing. But they are still not using the computer as an educational tool, except perhaps for general "computer literacy" instruction. In a recent survey of Texas Instruments and IBM microcomputer users, for example, not one person indicated that he or she used educational software (Duffy and Kelly, 1985). Publishers still consider educational software to be a high risk investment. Even in the American military, where much of the basic research on CBI has been conducted, computers seldom see active duty in the classroom.

A major reason for CBI's limited success is that producing high quality materials is

---

[1]Portions of this paper were presented at the NATO Symposium on Computer-based Instruction in Military Environments, Brussels, Belgium, April, 1985.

difficult and expensive. Developing dynamic, highly interactive instruction requires considerable programming skill as well as substantial instructional planning. The designer must understand not only instructional principles, but also the strengths and weaknesses of computerized delivery. He or she must integrate research on everything from screen displays to interface design in order to produce successful CBI. The combined programming and design effort adds significantly to production costs. Producing just one hour of high quality CBI demands 200 to 400 hours for TICCIT, 77 to 714 hours on PLATO, and 475 hours for the IBM 1500 Instructional System (Orlansky and String, 1979; Fletcher, 1984).

Cost is only one problem for CBI development, however. Another is the need for many kinds of expertise in the design process. We have only recently begun to train enough instructional designers and to develop strategies for a coordinated team approach involving both the designer and the subject matter expert. Now, with CBI, still another kind of expertise is necessary: computer programming. Too often, in fact, programming takes the spotlight in CBI development and overshadows concern for the quality of instruction, including effective sequencing and presentation strategies (Merrill, 1984; Regeiluth, 1984). Typical CBI is an unimaginative collection of drill-and-practice exercises, often preceded by on-screen "pages" of textual instruction. While the programs presented in other chapters of this volume are innovative and exciting, they are certainly the exceptions, not the rule.

In the last few years, several authoring tools have appeared in an attempt to reduce production costs and return the focus of development to instructional design. A sampling of such tools is described in a recent issue of the *Journal of Computer-Based Instruction* (1984, #3). These tools are generally high-level authoring languages which include special commands relevant to instructional programming--commands, for example, to create branches or check answers. Their main purpose is to facilitate the coding

process: insofar as they as successful, they reduce the emphasis on programming and leave more time for attention to instructional principles.

However, simple aids to coding cannot, in themselves, promote more effective instructional design. To achieve that goal, an authoring *system* should be devised which aids the entire instructional design process as well as coding. The United States Department of Defense is engaged in developing such a system, one which would not only simplify the production process but also encourage the principled design of instruction.

The success of an authoring system--or any computer system, for that matter--depends largely upon two things: the system's capacities or power and the user's ability to exploit those capacities. Computer systems are becoming more powerful as a natural consequence of technological improvements. Users, however, are not faring so well. Fischer, Lemke, and Schwab (1985) report that according to their data, users only take advantage of about 40% of a complex system's capabilities. Users are often unable or unwilling to explore the full powers of a system because they cannot find the information that would enable them to do so, either in the printed documentation or as part of an online help system. Indeed, the inadequacies of computer documentation and its often adverse effects on system use are legendary.

Developers of authoring aids are beginning to recognize that making their tools easier to use is an important goal. Fairweather and O'Neal (1984) describe recent authoring tools and systems as not only very powerful but also ". . . menu driven, highly prompted, and completely helped," suggesting the new emphasis on usability. However, the help in these authoring tools is usually restricted, consisting of a page or so of help information on each command describing its format and function. In other words, it is typical of the help provided by most computer systems, perhaps best exemplified by the UNIX help facility (Norman, 1982).

In the remainder of this chapter, we review the research and present the issues involved in designing effective online help systems. Of necessity, the review and much of the discussion focus on broad considerations for help systems, issues that are relevant to any application software. At various points, however, we look specifically at what the research implies for authoring languages or systems.

Defining the Audience

The general goal of a help system, regardless of application, should be to provide the typical user with all the information he or she needs to efficiently operate the system. Furthermore, help should be available without disturbing or limiting the user's activities. The goal cited by Fenchel and Estrin (1982) seems especially appropriate: the optimal help system will allow any application with which the user interacts to describe itself.

Historically, the design of help systems has focused more on avoiding disruption than on providing complete information, mainly because until recently most users were highly trained computer specialists. Such users required only minimal help since they had broad experience and were surrounded by other highly trained users. If the help information was incomprehensible, a user could simply ask a colleague. Under these conditions, even the most inadequate help system is tolerable as long as it is unobtrusive. Draper (1984) found, for example, that the UNIX "man" and "key" system is easy to use for those with considerable UNIX experience.

However, thanks to the increased availability and affordability of computers, a growing number of users are just learning about computer systems. The problems of these "naive" or "novice" users have led to a more widespread interest in the role and the design of help systems. One response to the naive user group has been the development of online introductory tutorials such as Apple's "Apple Introduces Apple." Most such tutorials cover initial user activities. While they provide users with some

hands-on experience, they seldom aid with anything beyond the basics. Therefore, they are not likely to assist users in exploring the system's full capabilities.

Today, most developers of help systems have the naive user in mind during the design process. However, another equally important and rapidly-growing user group has appeared: the computer sophisticated user who is naive only to the particular application program or the subject matter for which the software was designed (accounting in the case of spread sheets, for example). Whereas the novice user arose out of the general proliferation of computers, the "experienced novice" is a product of the proliferation of application software. An experienced novice is familiar with computers and with some software, and so has little need for the basic introductory tutorials. However, he or she needs more than the rudimentary command-level descriptions that might suffice for computer programmers. The experienced novice is not a programmer; he or she is interested in using computer resources to solve a particular problem in a discipline other than computer science. For example, the user of authoring software might be either a subject matter expert or an instructional designer who wishes to use the computer as a tool to produce educational materials.

The user of widely-varying application programs, whether a novice or an experienced novice, must have two kinds of information at hand. First, he or she must learn the specific functions and command structure of each piece of software. Second, as application programs become more sophisticated and offer more support for complex problem-solving in specialized areas, the user will have to understand the major strategies of problem representation and problem solving within each domain.

An adequate help system for a sophisticated application program must provide assistance with both kinds of information. For an authoring system, such help would include not only command level explanations, but guidance with strategies for more macro design tasks (designing the assessment, for instance, or linking presentation and

practice) and help in understanding the instructional model underlying the system. That is, if the software is a tool to aid authoring, then the help must extend beyond the command level to authoring tasks and the instructional design model. The subject matter, not just the program itself, must become a factor in creating the help.

The changes in the user population and the software market raise new issues for the design of online help systems. After a brief survey of experimental help systems, we will examine several of these issues in the remainder of this chapter and conclude by presenting recent research findings which suggest that the design of the information contained in the help system is a critical concern.

Prototype Help Systems

While designers and computer companies are recognizing the importance of helping the user and virtually every commercial program has some sort of help system, there is a paucity of research on the design of help systems. A number of individual researchers have done general surveys of the field (Houghton, 1984; Sondheimer and Relles, 1982; Relles and Price, 1981; Relles, Sondheimer, and Ingargiola, 1981) or have worked on relatively isolated aspects of help systems (Carroll and Kay, 1985; Pollack, 1985; Fenchel and Estrin, 1982; Rich, 1982, Shrager and Finin, 1982; Temin, 1982). Several sustained research efforts into basic human factors issues relevant to online help have come from Don Norman and the User Centered System Design Group at the University of California (Norman and Draper, in press); Ben Shneiderman and his colleagues at the University of Maryland (Shneiderman, 1984; Shneiderman, 1980); and Phil Hayes and the COUSIN interface group at Carnegie-Mellon University (Hayes, 1982; Glasner and Hayes, 1981; Ball and Hayes, 1980).

Much of the research to date has been concerned with the development of experimental prototypes. Specifically, many of the development efforts have been channelled into the following nine prototype help systems:

1. a help system for the SIGMA message system. Designed by Rothenburg (1979, 1975) at USC, this is perhaps the most complete and most sophisticated help system proposal, providing dynamic access to the help data base, help based on a user profile and linked to the command line interpreter, both interactive and textual tutorials, and integrated on and offline help.

2. Thumb, a system for locating materials within texts such as online manuals. Developed by Price (1982), Thumb creates a "passage tree" or functional outline of an online document, allowing the user to locate and retrieve only those passages relevant to his or her needs. Thumb aids the user by providing extensive cross-referencing and access tools. Prototypes implemented at NASA and BNR.

3. a help system for SARA, a software engineering tool. Developed by Fenchel and Estrin (1982), the system provides pointers to the help database according to command line input, responds to syntactic errors, provides further detail on request, and simplifies the entry of help information by the programmer of a new application.

4. a help system for Consul, an experimental message system. Developed by Mark (1982) at USC, this system is model-driven, responding to help requests by consulting online models of user behavior and system capacity, and mapping between the models. It accepts natural language input and responds with assistance to any system command that cannot be executed. It provides a system acquisition aid to assist programmers in providing help information on each new application.

5. a help system for the COUSIN interface. Developed by Hayes (1982) at Carnegie-Mellon University, this system is based on the ZOG menu system, providing static and dynamic help generated from online descriptions of each tool documented, which are provided by the tool designers.

6. DOCUMENT, an online documentation system to support the full spectrum of programs available through the National Magnetic Fusion Energy Computer Center. Developed by Girill and Luk (1983), this system is noted for the sophisticated methods of accessing the help data base. The DOCUMENT system allows full integration of on and offline materials, contains an acquisition aid for documenters to insure consistency across help texts, responds to omitted parts of a command line with appropriate options, and varies the level of presentation according to user profile information.

7. an experimental help system for UNIX print commands developed by the UCSD group at the University of California (Smolensky, Monty, and Conway, 1984). They have examined the use of a quick-reference facility based on a task-oriented documentation format with this system.

8. the Document Examiner, an online reference manual. Developed by Walker (1985), the Document Examiner includes dynamic cross referencing, a system of "bookmarks" to help users return easily to previously-examined sections of the document, an advanced screen display strategy using three windows, and complete integration of online and offline documentation.

9. an experimental help system for UNIX directory and file commands developed by Borenstein (1985) at Carnegie-Mellon University. Borenstein's three-window display includes a separate help menu window that is constantly updated in response to the user's activities and a text window which allows the user to view the help information without sacrificing his or her current context (an incomplete command line, for example).

Unfortunately, few of these prototypes or their design features have been formally evaluated. Therefore, while the developers' conceptions of effective help have been embodied, we have little evidence of how effective or valid the underlying concepts are. The primary focus has been building the prototypes based on a best guess about proper responses to the major issues. These issues or factors are

1. the degree to which the help is dynamic or based on an analysis of the context;
2. the mechanism for accessing help;
3. the structure of the database; and
4. the amount and kind of information presented to the user.

Before considering these factors, we note that despite the general lack of empirical evaluation, the prototypes reflect an encouraging concensus about the minimal components of an optimal help system. Designers seem to agree that every help system should contain features reflecting five basic capabilities:

- Static help: the basic query-driven help. The user requests help via a command or menu and receives the appropriate textual information.
- Dynamic help: context-sensitive help. The user encounters a problem or is uncertain how to complete an activity, and receives information based on the current state of his or her interaction with the system.
- Tutorials: for new users of an application. Many current online tutorials are simply textual explanations of the facilities, though there is a growing interest in interactive tutorials.
- Error handling: automatic calls to the help system on the presence of an error condition or the detection of inefficient user operations, such as using several commands where one would suffice.

- Integration of online and offline documentation: the ability to use the help files to produce the offline manual, enabling the writer to update both at once.

Factors in the Design of Help

Dynamism

The "dynamism" of a help system is the degree to which it can assess and respond to the user's needs of the moment. Dynamism, sometimes referred to as "context-sensitivity," is a continuum along which help systems can range. At the simplest level is the traditional, static help system in which the user must fully specify the help topic through a query or menu selection. At the other end of the continuum is an intelligent system which possesses both a model of the user and an expert model, and can fully diagnose the user's problem. Between these extremes are systems in which help is presented to the user based on the system's analysis of various features: the current command line, the prior command sequence, or a profile of the user's prior experience (see Rothenberg, 1979 and Mark, 1982).

The kind of online assistance most widely available is the lower end of the continuum or static help, which in its most primitive form is simply an online manual accessible by keywords or menus. Static help is particularly useful for users who want to expand their knowledge of the system or are looking for information on a specific command, and is generally considered the minimum for an online help system. Its major weakness is its inability to offer assistance directly related to the present state of the user's activity. The user must already know what help he or she needs or engage in a trial-and-error process to find out.

A second weakness is that the system cannot select in advance the category of help information needed--examples, say, or command syntax--or offer alternative strategies. The system does not "know" what the user is doing and hence will usually provide an

information dump on one command along with pointers to the most closely related commands. These two weaknesses are, of course, the classic problems with the UNIX help system.

Reliance on static help alone will be inadequate for an authoring system to be used by both subject matter experts and instructional technologists. We cannot assume that either group will be familiar with the particular instructional model and strategies for design that the system embodies, so the static help should be augmented by some form of dynamic help. Dynamic help occurs when at least some portion of the current context is used to guide the help response. It attempts to provide the information most likely to aid the user at the moment, together with pointers to further information. Dynamic help can be provided from the same database as the static help. However, the ideal system will reconfigure the presentation of help information, particularly the help categories and the degree of explanation, according to its analysis of the user's needs.

As we move along the continuum toward more dynamic help, we engage in a tradeoff between the inferencing supplied by the user and that supplied by the help system. Dynamic help at any level is based on some amount of inferencing about what the user wants to do and because of that will be subject to errors of inference. Help based on a mistaken inference can be as bad as no help at all. The user must spend valuable time trying to escape from unwanted or inappropriate help before he or she can locate the needed help. The likelihood of inferential errors increases as we attempt to make the system more intelligent.

Many designers agree that the minimal amount of dynamic help would allow the user to engage the help system at any point in the interactive dialogue and receive a brief explanation of his or her options at that point (Relles and Price, 1981; Relles, Sondheimer, and Ingargiola, 1981; Wasserman, 1981). At least this kind of dynamic help should be available for an authoring aid. Such a capacity requires that the help system be supplied with system state information and some record of a user's command history.

While assistance with the user's next possible commands seems a feasible level of help for an authoring system, moving any further toward the AI end of the continuum seems unwarranted except for experimental purposes. We do not know enough about the instructional design process to provide the system with an adequate basis for effective inferencing. The development of an intelligent system would require both an expert model and a model of the user. The system must be able not only to detect errors or suboptimal strategies (that is, deviations from the expert model), but also to infer the faulty reasoning that led to the error by consulting a user model in order to supply the appropriate help.

At the present time, we simply do not have a user model, or even data which could be used to generate one. We could not locate any data on what we consider to be two critical, minimal components of a user model: 1) assessing and contrasting the goals and strategies of the two primary user groups (subject matter experts and instructional technologists) and 2) evaluating expert and novice behavior in each group.

Access

Current help systems use three basic mechanisms for accessing help information. First, the help options may always be visible on the screen. For static help this would be a menu based system best characterized by ZOG and COUSIN (see Hayes, 1982). For dynamic help systems, the help would also be in a menu display but available in a separate window and constantly updated in response to the user's choice of commands. The WordStar help menu is perhaps the best available example of this dynamic, visible help (*WordStar Reference Manual*, 1983). In general, designers assume that this visible presentation of help options is best for novice users. When a novice is first using WordStar, for example, he or she can always have available the list of next possible commands. Then, as expertise develops, he or she can hide the help selections and make more of the workspace available for composing.

The second means of accessing help is through a query system. In this case, help options are not presented on the screen; rather, the user must request help explicitly. For static help systems, the user must request information on a specific command as in the UNIX "key" and "man" systems. For dynamic help, the user simply requests that the help options be presented. The same information constantly available in the "visible" help system is available only on request. Such a "hidden" menu system is exemplified by the TOPS-20 system used at CMU (Bond et al., 1982).

Finally, the system may present help when it infers that the user needs assistance, usually by recognizing that a system error has occurred. Error handling by online help systems requires, at a minimum, that a trigger to the help system be attached to the machine state or the command interpreter, so that the presence of an error condition or a command which cannot be executed generates a call to the help system. Both Rothenberg's (1975) Tutor and Fenchel's help system for SARA (Fenchel and Estrin, 1982) propose such a mechanism. Ideally, the system would also contain a history of the user's activities so that it can determine how much help the user probably needs. Rothenburg's system also contains such a user profile.

The availability of powerful multi-window computing environments with varying access devices (icons, mice, etc.) makes the issue of user access mechanisms much more complex than a simple three-alternative, keyword-menu-automatic choice. To our knowledge, no research has been done exploring the implications of these new technologies for user access. Indeed, only Rothenberg (1979), Borenstein (1985), and Walker (1985) have employed advanced displays and windows, but none reports any testing of alternative access or display strategies based on the technology.

Information Requirements

Level of Explanation

Most help systems, regardless of design, only provide help with individual commands, usually in the form of an online reference manual. While this may be adequate for operating systems, where the application goals vary widely, it appears completely inadequate for targeted application programs. Authoring languages, for example, are meant to aid users in creating dynamic and highly interactive instruction. Yet typical help systems would not provide assistance in integrating a series of commands to accomplish that goal. The help is on individual commands, not on strategies for using the commands in a larger context. As we mentioned earlier, even if an authoring language advertises the availability of menu-driven help systems, the commands on the menus tend to be at the micro editing level and the author is left to his or her own devices to determine strategies for moving through these commands.

We have argued that the help system must provide at least three "levels of explanation" in an authoring system or other sophisticated application program. The lowest level is help with individual commands, which is almost always organized by the command structure, including the format and options. O'Malley et al. (1983) have proposed that even at the command level, the help information might be made more responsive to the the user's needs by organizing it around tasks the user performs (his or her goals) rather than the command set.

In fact, the UCSD group (Smolensky et al., 1983) has embodied a task-oriented approach in a command level help system for the UNIX print commands (printing tasks). Their "task attributes" include such categories as printing method, formatting, and portion of files printed. They recommend the development of an "attribute encyclopedia," which brings together under task-oriented headings all the information relevant to one aspect of

a task (formatting, for example). Their task-oriented help is based largely on re-organizing and re-formatting the command descriptions. For example, rather than listing the options for a command alphabetically, the task-oriented information lists the options by their functions. To our knowledge, the UCSD group has not tested this task-oriented format against a command-oriented format.

At the next level, the help must set the command into the context of the application's larger framework. In the case of an authoring system, the user is employing the individual command as part of the larger goal of creating a component of instruction. The help should relate the command to the design of the component. Such help might include, for example, strategies or examples of how to create different assessments, practice presentations, feedback, or animated segments (assuming, as seems reasonable, that completing each of these tasks will require a lengthy command sequence).

The third level of help information would assist the user in the instructional design process. Any system that aids instructional design will embody some model of the instructional design process and "good" instructional techniques. In an authoring language or system, the model will be reflected in the macro commands available (and not available), the ease of linking components of instruction (for instance, presentation, practice, and assessment), and the contingencies between the components and the overall instruction (that is, front end analysis, objectives, delivery, and assessment). The help system would simply make the model explicit, providing "system overview information" and instructional design advice. The model would show the relationships between the user's current task domain and other tasks or data bases in the authoring system. In a dynamic state, this overview information would also provide the author with a summary and an evaluation (relative to the model) of the instruction developed to that point.

Extensive evidence in the cognitive science and problem solving literature suggests that an individual's strategies for using a system are guided by an internalized model of that system. His or her effectiveness in problem solving will be largely dependent on how accurately that "mental" model corresponds to the actual workings of the system. Rumelhart and Norman (1981) have demonstrated the importance of such a functional model in the use of the UNIX screen editor. In one condition, they provided subjects with the reference information on the editor commands and examples of the application. Subjects in a second condition received the same information along with a description of how the text editor worked--that is, a model of the system. Subjects in the latter condition were significantly more successful in carrying out editing tasks.

Amount of Information Presented

One of the worst problems for users of static help systems is the deluge of information they often receive when they ask for help. The help system usually provides a large text containing much information that is irrelevant to the user's current needs. The user must scan through the text to find the target information, often a lengthy and frustrating process. Two solutions have been proposed to this problem: controlling the level of detail presented and controlling the kind or category of information presented.

Level of Detail

When the user wants only to check the syntax or options for a command, presenting an abbreviated form of the help information seems appropriate. O'Malley et al. (1983) at UCSD suggest that for these situations, two kinds of static help be available:

1. full explanation, which for any entry contains all the information presently available in the online manual; and
2. quick reference, which contains in each entry the correct syntax for the command, the options which may be supplied with it, and a brief explanation of its use.

The quick reference facility would also contain pointers to the full entries on relevant

topics. Bannon and O'Malley (1983) evaluated a quick reference facility on the UNIX system in their University of California research lab, finding that users liked it and made fewer calls to "man" for full explanations after the quick reference facility was implemented.

Categories of Information

An alternative strategy is to present only a particular kind of information on a command or task. That is, a help entry might contain many categories of information (command format, option definition, applications, cross references, examples, bugs, and others). Rather than providing a condensed version of all this information, we could allow the user to select the categories he or she needs, or have the system infer the appropriate categories. If the user only needs to understand the command format, for example, he or she could receive only that information.

One way to implement the control of information categories is to organize the help database on a "hypertext" model (Nelson, 1974). Hypertext-oriented databases consist of small, richly-interconnected pieces of information which can be accessed and combined in a variety of ways. The pieces of a help text (syntax summary, options, examples, others) can be stored as separate database items and accessed either individually or by a routine that formats them into a reference manual entry. The COUSIN help system (Hayes, 1982) uses such an organization, as does the prototype system developed by Borenstein (1985). Such a database facilitates cross-referencing while improving the likelihood that a particular help request will yield only the information required.

Tutorials

While help systems present many different categories of help information to a user, few systems provide what might be considered a critical help -- tutorials. Tutorials differ from other help categories because they involve interaction with the user. The user actually works through a problem or example using the computer. The ideal tutorial system would in fact allow the subject to use a real-world problem as the basis for the tutorial. Alternatively, the user could complete the tutorial in one window while he or she works on ongoing activities in another window.

Tutorials seldom appear in help systems, most likely because the best tutorials require interactive programs. A few interactive tutorials are available which introduce particular systems or software, such as "Apple Introduces Apple" and the Lotus 1-2-3 tutorials (Posner et al, 1983). While these products, as well as the tutorials found in most hard copy documentation, make clear the importance of tutorials, we have been able to identify only one experimental help system that included interactive tutorial assistance, the proposed Tutor module for the SIGMA message system (Rothenberg, 1975; 1979). We present a brief summary of its proposed features to suggest the directions that future development in interactive tutorials might take.

In Rothenburg's Tutor module, the online tutorials would perform three functions: 1) introducing new users to the operation of the system; 2) allowing users to expand their knowledge of system capabilities; and 3) intervening with assistance when the user seems to be attempting something he or she does not yet know how to do. The module was designed to combine textual with interactive tutorials. The user would read about the system briefly and do exercises with it, able to "try" certain procedures within a special protected environment which prevents the sending of practice messages and the alteration of data. Commands issued within the protected environment would be executed as far as possible without damage to data or the system state; the

consequences of the commands beyond that point would be simulated. The Tutor module would also provide an authoring language for creating interactive lessons.

An Evaluation of a Help System

Most of the work in designing help systems has focused on the issues of dynamism and access, while few have attended to the quality of the information presented. Providing increasingly dynamic help is a movement toward an intelligent system and many system designers, especially programmers, are interested in that goal. The access issue is nicely circumscribed and therefore readily amenable to human factors analysis. The quality of the information, by contrast, is difficult to assess and is not explicitly a computing variable, which may account for the sparse attention paid to it. However, a dissertation recently completed by Nathaniel Borenstein at Carnegie-Mellon University provides data suggesting that the quality of the information may be the critical variable in the design of a help system.

Borenstein (1985) designed a prototype help system called ACRONYM for the UNIX operating system. We will not consider the details of the database structure and access mechanisms here, but will focus on the human-computer interface which was a primary concern of Borenstein's evaluation. For the prototype system the screen was divided into three parts. The command line window was in the bottom portion of the screen; a menu of help options appeared in the middle of the screen; and a brief help text could be presented in the top part of the screen. Of the different experimental conditions described below, only ACRONYM contained the multi-window display.

Two help access conditions appeared in the evaluation. In the first, a context-sensitive condition, the command line was parsed and the menu and text displays were modified to reflect the options available to the user at a given point in the command sequence. For example, if the user typed "rm," the options and arguments for "rm"

would be displayed in the top part of the screen and a menu of further information would appear in the middle portion, including "examples," "options," "delete a directory, rmdir," and "what is a file." In this condition, the user could also request help for a particular command by using "man" or "key" in the bottom command line. The second help condition was the traditional UNIX static help system. That is, the user could only receive help by requesting it via "man" or "key."

A second variable was the help text presented to the user. In one condition the text was the standard UNIX "man" text in which commands are described in terms of the name, format, description, cross references, bugs, and relevant system files. The function, options, and any examples of the command are all presented in the description section. The alternative text condition was basically the UNIX help information presented in Sobell's (1984) *A Practical Guide to the UNIX System*. The alternative help text, then, was not idealized nor was it customized for this particular study. However, it was in general more clearly written than the "man" text and was divided into more even and more functional chunks of information. For example, the command summary, options and examples each appeared in a separate section. In contrast to the standard UNIX manual entries, there was always an example. Information beyond the basic description was presented in a separate "additional notes" category.

Borenstein combined these two variables to create three experimental conditions: standard UNIX (static help and UNIX documentation); simplified text (static help and the Sobell text); and ACRONYM (both static and dynamic help, the Sobell text, and the three-window display). A fourth condition was meant to set the standard for the maximally effective help: a human tutor condition in which subjects could ask the tutor any question, but the tutor could not use prior knowledge of the tasks to respond.

All of the subjects were experienced computer users; however, one group was experienced with UNIX while the other was not. The subjects were given 12 tasks to

complete using the standard UNIX help system and 12 tasks to complete using one of the three experimental help systems. Experienced and inexperienced users received different tasks reflecting their different levels of expertise.

The average time per task for experts and for novices is presented in Table 1. When entered into a regression analysis to remove task and subject variability, the data for experts and novices supports essentially the same conclusions. The simplified text made up about half the difference between the standard UNIX help and a human tutor. The provision of dynamic help, with its menus and command summaries, did not seem to aid performance over and above the simplified text. There was no performance difference between the simplified text and the ACRONYM conditions for either the experts or novices, though subjects did show an affective preference for the ACRONYM condition, perhaps because the display allowed them to continue working while help information appeared in a separate window.

Borenstein's results indicate that the quality of the help text is far more important than the mechanism by which the help text is accessed, and this conclusion holds for both experts and novices. The data do not indicate the nature of the "quality" variable that led to the significant effects. However, the effects apparently are not due to simple readability. Borenstein found the readability of both Sobell (1984) and "man" texts to be at approximately the 10th grade level according to several standard readability formula. More likely "quality" variables are the action-oriented style of writing; the use of meaningful headings and the more even chunking of information; and the more performance-oriented contents of the information presented.

Summary

Throughout this chapter we have emphasized the importance of a multi-level help system that goes beyond simple command descriptions to include assistance with goals and strategies within the application domain--in this case, automated authoring. The help system should be dynamic in at least some degree, basing its presentation of help options on the user's current activity. We would also suggest that an adequate help system would include a mechanism for controlling the amount of information presented on a help request and an interactive facility to allow users to try out different features in a procedural fashion. Finally, the textual information contained in the help database must be carefully designed and structured to make the necessary assistance both accessible and understandable for maximum user efficiency.

Table 1

Mean normalized time (sec.) to complete a task.
(From Borenstein, 1985)

	EXPERT	NOVICE
STANDARD UNIX	168	167
SIMPLIFIED TEXT	116	115
ACRONYM	139	103
HUMAN TUTOR	103	60

Note: The experts and novices received different tasks.

References

Ball, E. and Hayes, P. Representation of Task-Specific Knowledge in a Gracefully Interacting User Interface. In *Proceedings, First Annual National Conference on Artificial Intelligence*, Stanford University, 1980, 116-120.

Bannon, L.J. and O'Malley, C. Problems in Evaluation of Human-Computer Interfaces: A Case Study. In *User Centered System Design: Part II. Collected Papers from the UCSD HMI Project*, La Jolla, CA: Institute for Human Information Processing, University of California, San Diego, March 1984. ICS Report No. 8402, 67-73.

Bond, S.J., Hayes, J.R., Janik, C.J., and Swaney, J.H. (Eds). *Introduction to CMU TOPS-20*. Pittsburgh, PA: Communications Design Center, Carnegie-Mellon University, August 1982.

Borenstein, N.S. *The Design and Evaluation of On-line Help Systems*. Ph.D. thesis. Carnegie-Mellon University, 1985.

Carroll, J.M. and Kay, D.S. Prompting, Feedback and Error Correction in the Design of a Scenario Machine. In *Proceedings of the CHI 1985 Conference on Human Factors in Computing Systems* (San Francisco, April 14-18, 1985), ACM, New York, 149-153.

Draper, S. The Nature of Expertise in UNIX. In *User Centered System Design: Part II, Collected Papers from the UCSD HMI Project*, La Jolla, CA: Institute for Human Information Processing, University of California, San Diego, March 1984. ICS Report No. 8402, 19-27.

Duffy, T.M. and Kelly, P. *An Analysis of the Use of the Texas Instruments Inc. Professional Microcomputer*. Pittsburgh, PA: Communications Design Center, Carnegie-Mellon University, June 1985.

Fairweather, P.G. and O'Neal, A.F. The Impact of Advanced Authoring Systems on CAI Productivity. *Journal of Computer-Based Instruction*, 11 (1984), 90-94.

Fenchel, R.S. and Estrin, G. Self-describing Systems Using Integral Help. *IEEE Transactions on Systems, Man and Cybernetics*, smc-12, 2 (March/April 1982), 162-167.

Fischer, G., Lemke, A., and Schwab, T. Knowledge Based Help Systems. In *Proceedings of the CHI 1985 Conference on Human Factors in Computing Systems* (San Francisco, April 14-18, 1985), ACM, New York, 161-167.

Fletcher, J.D. Intelligent Instructional Systems in Training. In S.A. Andriole (Ed.). *Applications in Artificial Intelligence*. Petrocelli Books, Inc., 1984.

Girill, T.R. and Luk, C.H. Document: An Interactive, Online Solution to Four Documentation Problems. *Communications of the ACM*, 26, 5 (May 1983), 328-337.

Glasner, I.D. and Hayes, P.J. *Automatic Construction of Explanation Networks for a Cooperative User Interface.* Technical Report CMU-CS-81-146, Carnegie-Mellon University Department of Computer Science, November 1981.

Hayes, P.J. Uniform Help Facilities for a Cooperative User Interface. In *AFIPS Proceedings of the National Computer Conference*, 1982, AFIPS Press, Montvale, NJ, 469-474.

Houghton, R.C. Online Help Systems: A Conspectus. *Communications of the ACM*, 27, 2 (February 1984), 126-133.

Mark, W. Natural-Language Help in the Consul System. In *AFIPS Proceedings of the National Computer Conference*, 1982, AFIPS Press, Montvale, NJ, 475-479.

Merrill, D. Component Display Theory. In Reigeluth, C. (Ed.). *Instructional Design Theories and Models: An Overview of Their Current Status.* Hillsdale, NJ: Lawrence Erlbaum and Associates, 1983.

Nelson, T.H. *Dream Machines.* South Bend, IN: the distributors, 1974.

Norman, D.A. and Draper, S.W. (Eds.). *User-Centered System Design: New Perspectives in Human-Machine Interaction.* Hillsdale, NJ: Lawrence Erlbaum and Associates, in press.

Norman, D. The Trouble with UNIX. *Datamation*, 27, 12 (November 1981), 139-150.

O'Malley, C., Smolensky, P., Bannon, L., Conway, E., Graham, J., Sokolov, J., and Monty, M.L. A Proposal for User Centered System Documentation. *Proceedings of the CHI 1983 Conference on Human Factors in Computing Systems* (Boston, December 12-15, 1983), ACM, New York, 5-8.

Orlansky, J. and String, J. *Cost Effectiveness of Computer-based Instruction in Military Training* (IDA paper P-1375). Arlington, VA: Institute for Defense Analysis, 1979.

Pollack, M.E. Information Sought and Information Provided: An Empirical Study of User/Expert Dialogues. In *Proceedings of the CHI 1985 Conference on Human Factors in Computing Systems* (San Francisco, April 14-18, 1985), ACM, New York, 155-159.

Posner, J., Hill, J., Miller, S.E., Gottheil, E., and Davis, M.L. *Lotus 1-2-3 User's Manual.* Cambridge, MA: Lotus Development Corporation, 1983.

Price, L.A. Thumb: An Interactive Tool for Accessing and Maintaining Text. *IEEE Transactions on Systems, Man and Cybernetics*, smc-12, 2 (March/April 1982), 155-161.

Reigeluth, C. and Stein, F. The Elaboration Theory of Instruction. In Reigeluth, C. (Ed.). *Instructional Design Theories and Models: An Overview of Their Current Status.* Hillsdale, NJ: Lawrence Erlbaum and Associates, 1983, 335-379.

Reiles, N., and Price, L. A User Interface for Online Assistance. *Proceedings of the Fifth International Conference on Software Engineering*, 1981, 400-408.

Relles, N., Sondheimer, N.K., and Ingargiola, G. A Unified Approach to Online Assistance. In *AFIPS Proceedings of the National Computer Conference*. 1981. AFIPS Press, Montvale, NJ, 383-387.

Rich, E. Programs as Data for their Help Systems. In *AFIPS Proceedings of the National Computer Conference*, 1982, AFIPS Press, Montvale, NJ, 481-485.

Rothenberg, J. Online Tutorials and Documentation for the SIGMA Message Service. In *AFIPS Proceedings of the National Computer Conference*, 1979, AFIPS Press, Montvale, NJ, 863-867.

Rothenberg, J. *An Intelligent Tutor: On-line Documentation and Help for a Military Message Service*. Technical Report ISI/RR-74-26, USC-ISI, May, 1975.

Rumelhart, D.E. and Norman, D. Analogical Processes in Learning. In Anderson, J. (Ed.). *Cognitive Skills*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1981.

Shneiderman, B. *Human Factors Issues of Manuals, Online Help, and Tutorials*. Technical Report CS-TR-1446, Department of Computer Science, University of Maryland, September 1984.

Shneiderman, B. *Software Psychology: Human Factors in Computer and Information Systems*. Cambridge, MA: Winthrop Publishers, 1980.

Shrager, J. and Finin, T. An Expert System that Volunteers Advice. *AAAI-82*, (1982), 339-340.

Smolensky, P., Monty, M.L., and Conway, E. Formalizing Task Descriptions for Command Specification and Documentation. In *User Centered System Design: Part II. Collected Papers from the UCSD HMI Project*. La Jolla, CA: Institute for Human Information Processing, University of California, San Diego, March 1984. ICS Report No. 8402, 51-63.

Sobell, M. *A Practical Guide to the UNIX System*. Menlo Park, CA: The Benjamin/Cummings Publishing Company, 1984.

Sondheimer, N. and Relles, N. Human Factors and User Assistance in Interactive Computing Systems: An Introduction. *IEEE Transactions on Systems, Man and Cybernetics*, smc-12, 2 (March/April 1982), 102-107.

Temin, A.L. *The Question-Answering Module in an Automated Natural Language Help System for the Text-Formatter Scribe*. Thesis proposal, University of Texas at Austin, 1982.

Walker, J. Implementing Documentation and Help Online. Tutorial conducted at the 1985 Conference on Human Factors in Computing Systems, April 14-18, San Francisco, CA.

Wasserman, A. User Software Engineering and the Design of Interactive Systems. In *AFIPS Proceedings of the National Computer Conference*, 1981, AFIPS Press, Montvale, NJ, 387-393.

WordStar Reference Manual (Release 3.3). San Rafael, CA: MicroPro International Corporation. 1983.